Compilers Lab Final

True / False

a. Symbol Table is needed only in Analysis Phase
- False

b. Compilers can transform from Source Code to Assembly Code.
- True

c. Linkers map all the library functions with their corresponding headers of library file.
- True.

d. Interpreters gives us better error ~~diago~~ diagnosis
- True

e. Synthesis Phase builds Syntax tree.
- False

f. Optimizer is one of the most important Phases of Compilers, hence must be implemented.
- False

g. Compilers can transform from Source Code to machine Code alone.
- False

h. Compilation gives better errors report.
- False

i. Inherited attributes take siblings value.
- True.

j. Bottom up tree approach is alone enough to handle the total grammatical expression.
- False

k. Postfix notation is used to evaluate the value of a statement.
- True.

l. <= is a lexeme
- True

m. "Increment" it is a Pattern.
- True

n. Lexeme $\xrightarrow{\text{Grammar}}$ Token
Is it true?.
- True.

o. Front end of a Compiler has 3 steps.
- True.

P. Compilers works in two Phases. - True

q. Semantical rule finds the meaning of a line.
- True.

r. Three address code is better to represent intermediate representation. - True

s. Non-terminal starts the derivation of a string.
- True

t. Analysis & Synthesis are the main two Parts of a Compiler. - True.

u. Compiler can be divided into 4 Categories.
- True

v. Syntax Analyser takes groups tokens of Source Program into grammatical Production
- True.

w Parsers are expected to Parse the whole Code
- True

x. A grammar for a Programming Language is a formal description of Structure.
- True

1. Compiler Can Check
   i. Logical Error
   ii. Syntax Error
   Which one is true?
   Ans: ii.

2. High Language → Interpreter → Lower Level Language
   → Assembly Language
   → Binary Language.
   Is it true?
   → Ans: True.

Interpreter generates intermediate object Code - False

#

Short Questions.

a. What is the run time of Insertion in a Symbol Table?

Ans: O(1)

b. Why we use delim?

Ans. To mark the beginning of the rule.

c. What is Yacc?

Ans: Yet another Compiler Compiler

d. How Yacc works?

Ans Look ahead Left to right Parser generator.

e. What is bison?

Ans: Parser generator

f. Write Command to run a flex.

Ans:

```
flex -t a.l > a.c
g++ -c -o a.o a.c
g++ -o a a.o -ll
./a
```

g. Write the role of lexical analyzer

Ans:
a. Identify tokens.
b. Remove whitespace
c. Install lexeme in symbol table
d. Return token to parser.

h. Write regular expression of viu id.

Ans:

~~all [0-9] [~~

011 1 [0-9] [1-9] 0 [0-9] [1-9]

i. Consider year range 1999 ~~-2050~~

j. Write regular expression of Date. [dd-mm-year]

Ans:

(0[1-9] | 1 [0-9] | 2 [0-9] | 3[01]) - (0 [1-9] | 1 [012]) - (1999)

g. Write code to run a bison file

Ans.

```
flex  calc.le
jacc  -d  calc.y
g++   -c  -w  lex.yy.c  y.tab.c
g++   -o  mycalc  y.tab.o  lex.yy.o
./mycalc
```

K. What is the Purpose of jjdebug?

Ans: It keeps the trace of Parser.

1. Suppose you have the given grammar

P → P Stmt | P ch = stmt | ε

Stmt → expn | ε

expn → expn + expn

| expn - expn

| expn * expn

| expn / expn

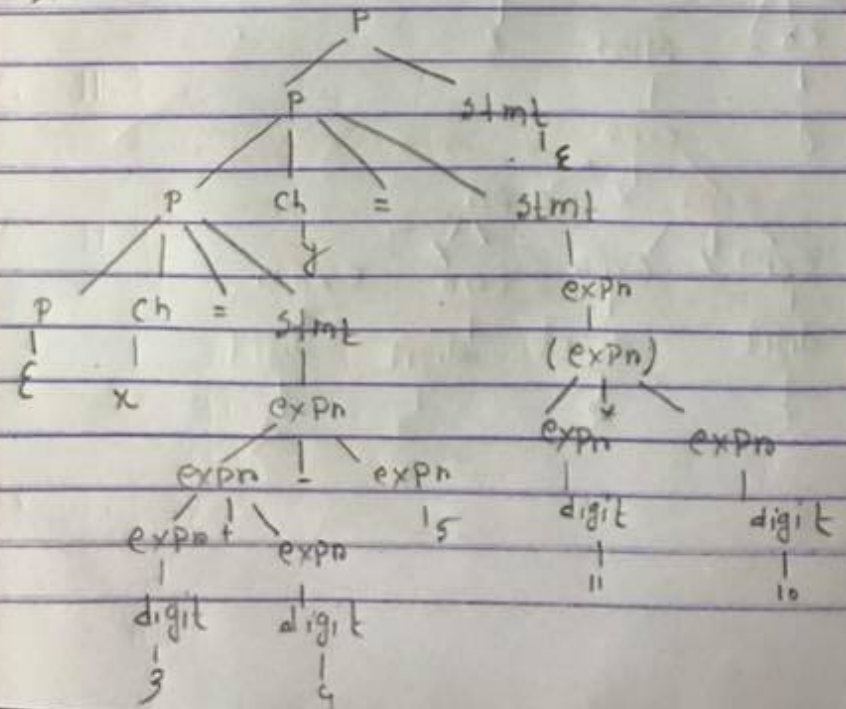
| (expn)

| digit | ε

x = 3 + 4 - 5

y = (11 * 10) => Draw Parse Tree

Ans:



7.

8

m. Look at the grammar

~~P →~~ P → P stmt | P ch = stmt | ε

~~Stmt → ε~~ stmt → expr | ε

expr → expr + expr

| expr − expr

| expr * expr

| expr / expr

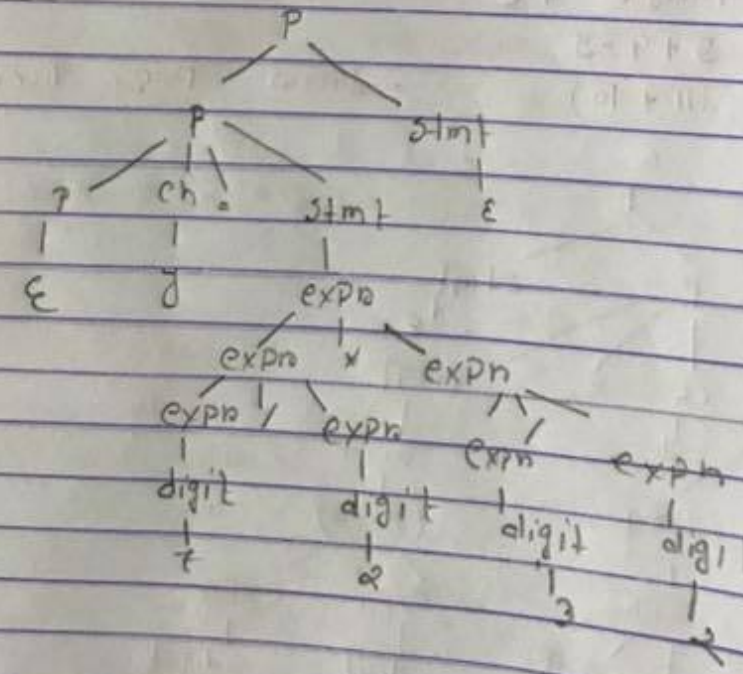
| (expr)

| digit

| ε

For z = 7/2 * 3/2 draw Parse tree.

Ans:



n. What extern means?
Ans: Compiler allocates memory outside the module.

o. What is the PurPose of yyleng?
Ans: Holds the length of the lexeme

P. What is the PurPose of yyin?
An: It is the input stream ~~Pointer~~ Pointers.

q. Write regular expression for integers number. Suppose, you have a integers number. You want to Check it is odd or even. Write code Segment.

Ans:

```
0|[1-9][0-9]*  { check (yytext); }

void yy check ( Char a[]) {
        int val = atoi (a);
        if (val %.2 == 0){
                Printf ("Even \n");
        }
        else{
                Printf ("Odd \n");
        }
}
```